Fig. 1

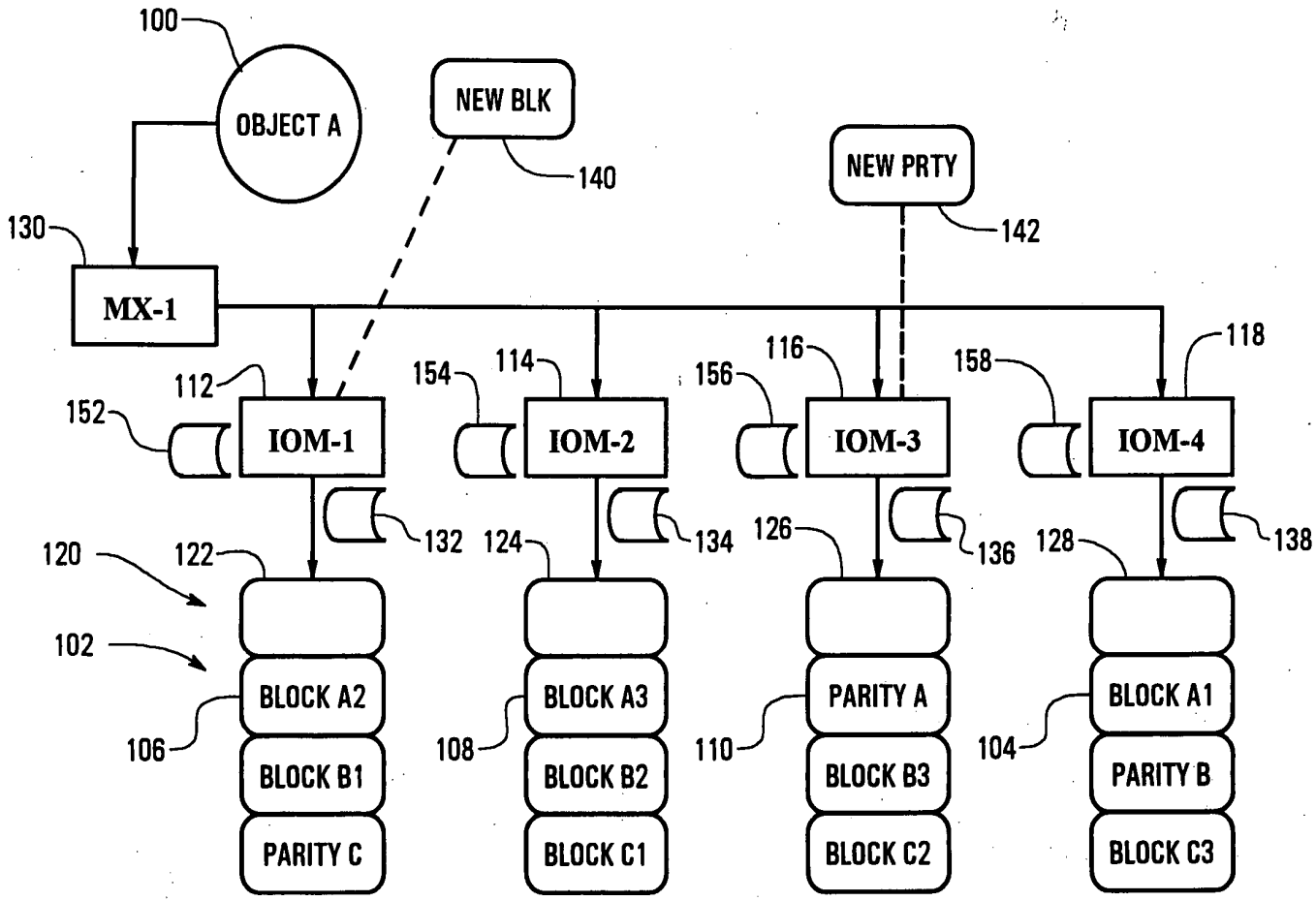

100
OBJECT A
NEW BLK
140
NEW PRTY
142
130
MX-1
112
152
IOM-1
154
114
IOM-2
156
116
IOM-3
158
118
IOM-4
132
124
134
126
136
128
138
120
122
102
BLOCK A2
106
BLOCK B1
PARITY C
BLOCK A3
108
BLOCK B2
BLOCK C1
PARITY A
110
BLOCK B3
BLOCK C2
BLOCK A1
104
PARITY B
BLOCK C3

Fig. 2

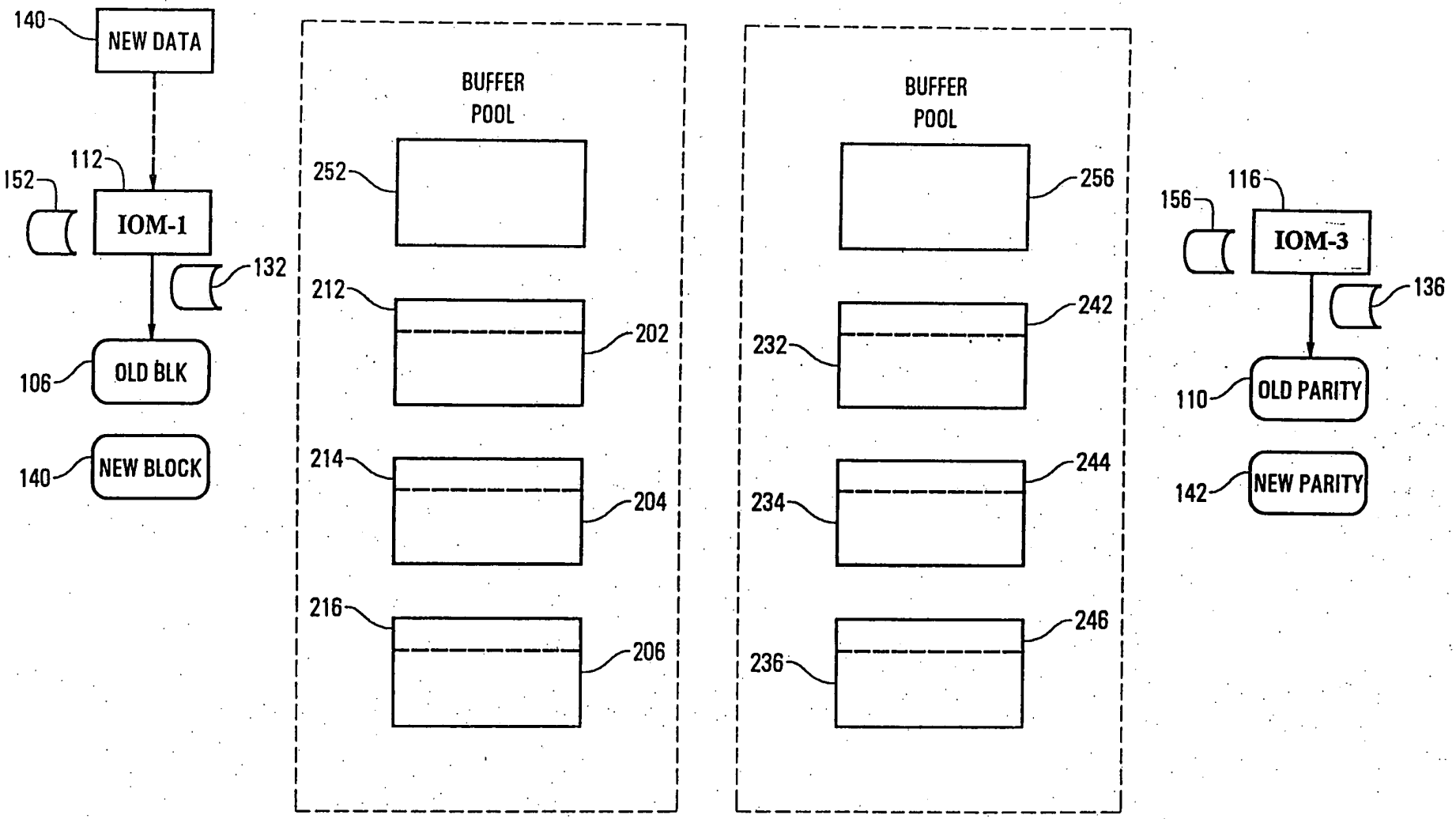

140
NEW DATA
112
152
IOM-1
132
106
OLD BLK
140
NEW BLOCK
BUFFER POOL
252
212
202
214
204
216
206
BUFFER POOL
256
242
232
244
234
246
236
116
156
IOM-3
136
110
OLD PARITY
142
NEW PARITY

Fig. 3

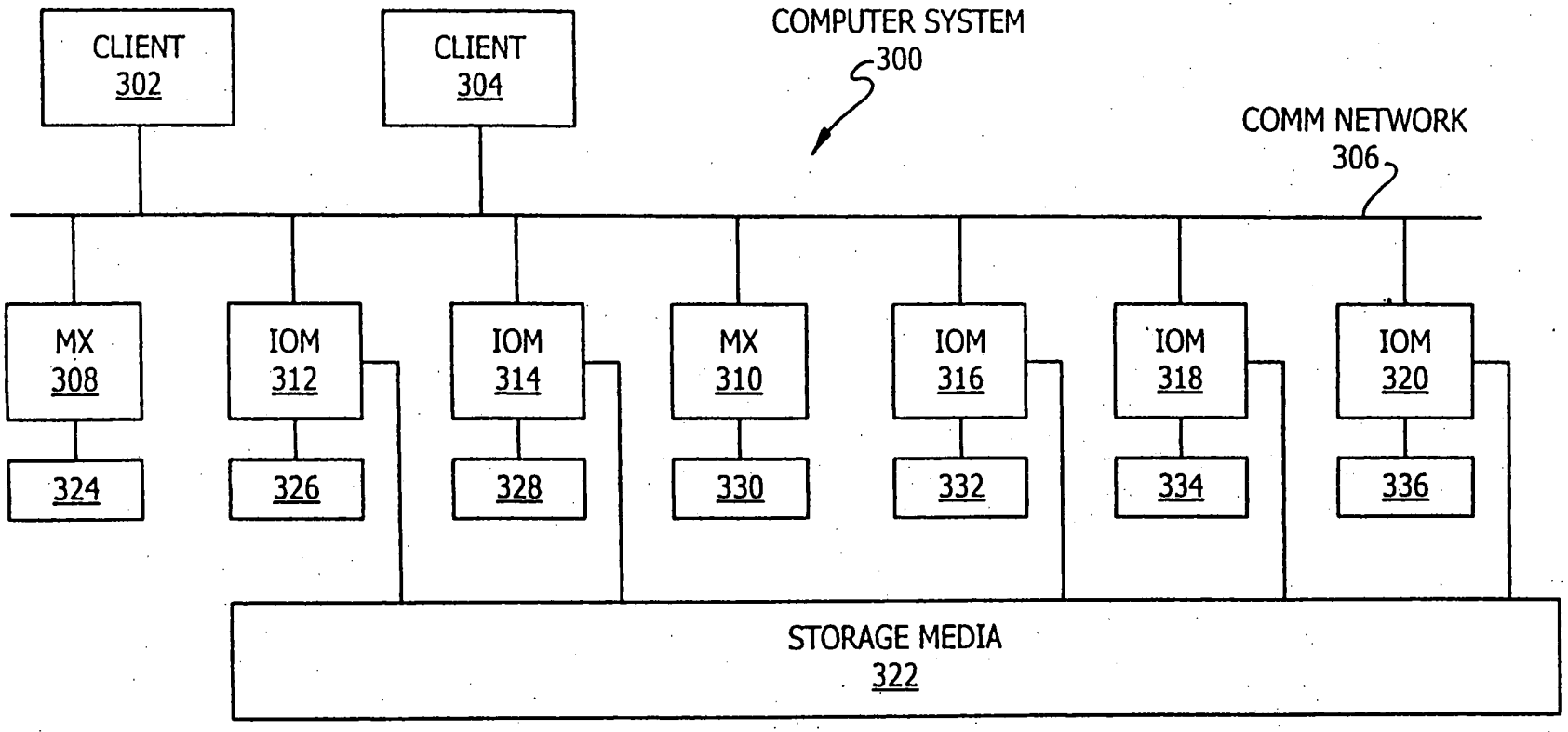

CLIENT
302
CLIENT
304
COMPUTER SYSTEM
300
COMM NETWORK
306
MX
308
IOM
312
IOM
314
MX
310
IOM
316
IOM
318
IOM
320
324
326
328
330
332
334
336
STORAGE MEDIA
322

Fig. 4

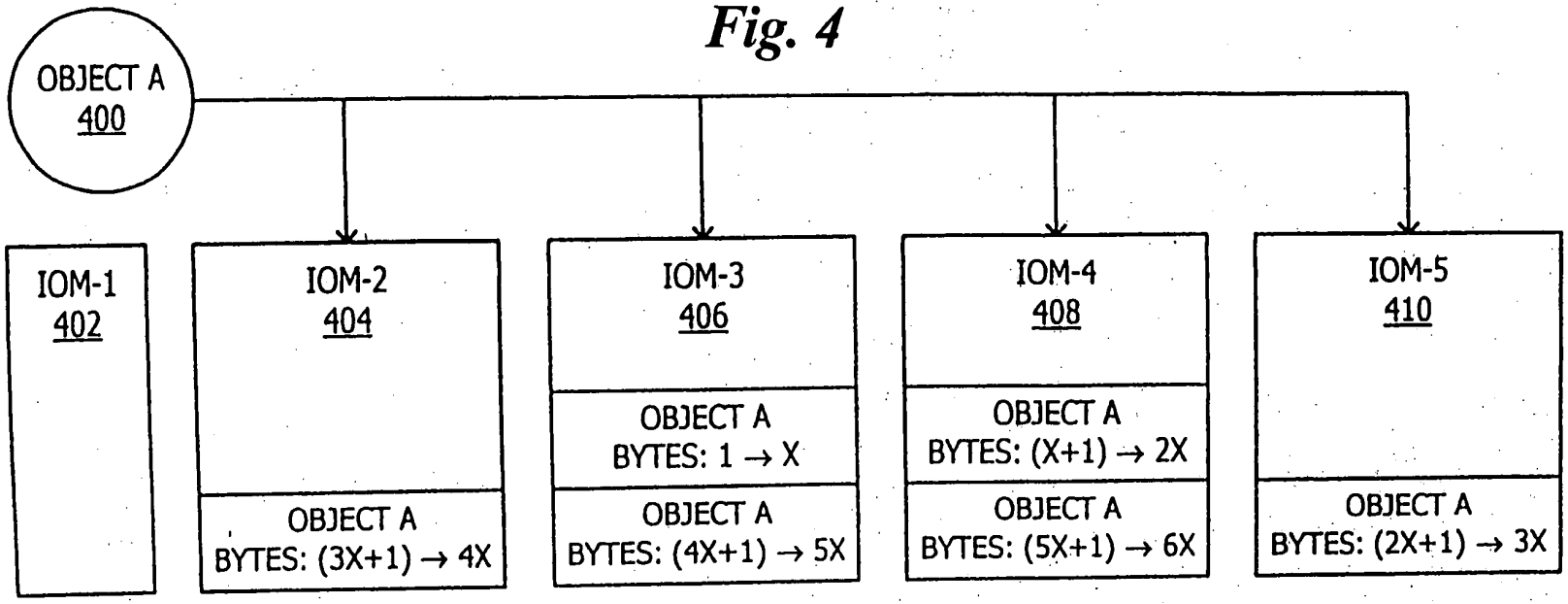

OBJECT A
400
IOM-1
402
IOM-2
404
OBJECT A
BYTES: (3X+1) → 4X
IOM-3
406
OBJECT A
BYTES: 1 → X
OBJECT A
BYTES: (4X+1) → 5X
IOM-4
408
OBJECT A
BYTES: (X+1) → 2X
OBJECT A
BYTES: (5X+1) → 6X
IOM-5
410
OBJECT A
BYTES: (2X+1) → 3X

Fig. 5

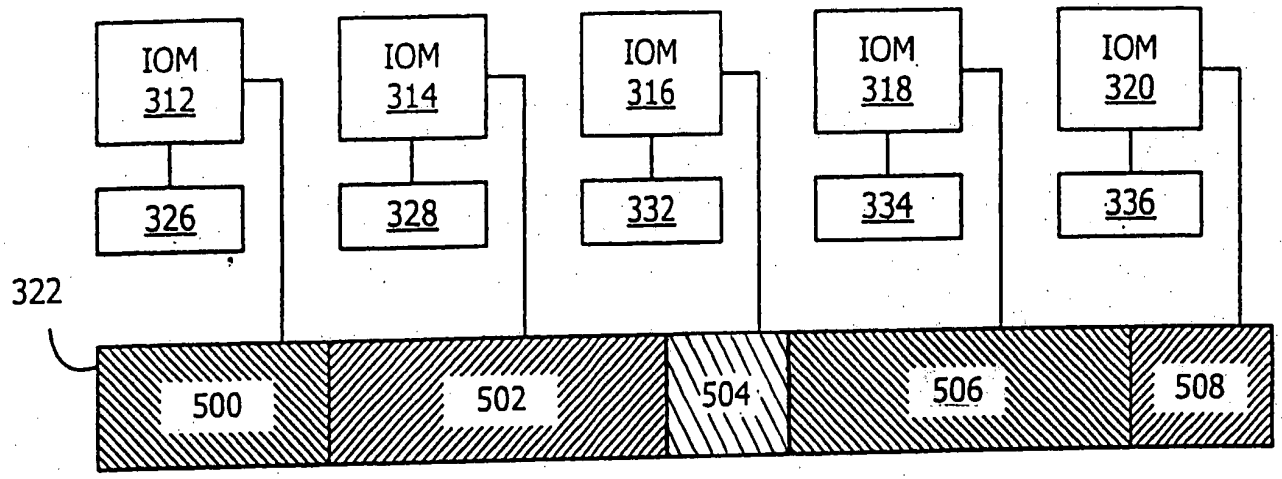

IOM
312
IOM
314
IOM
316
IOM
318
IOM
320
326
328
332
334
336
322
500
502
504
506
508

Fig. 6

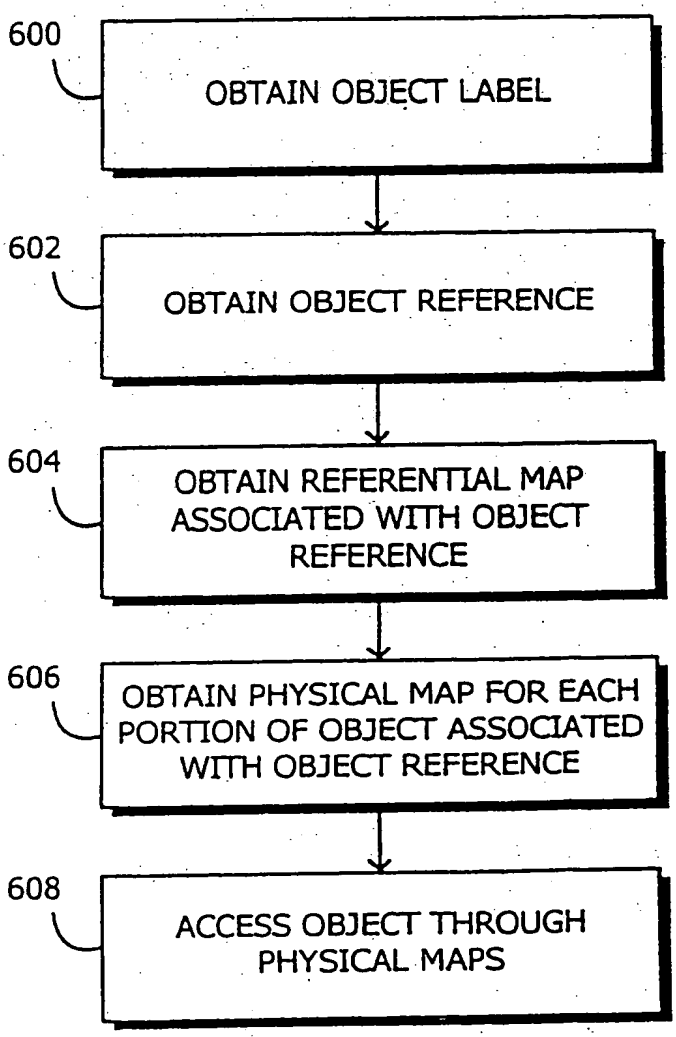

600
OBTAIN OBJECT LABEL
602
OBTAIN OBJECT REFERENCE
604
OBTAIN REFERENTIAL MAP ASSOCIATED WITH OBJECT REFERENCE
606
OBTAIN PHYSICAL MAP FOR EACH PORTION OF OBJECT ASSOCIATED WITH OBJECT REFERENCE
608
ACCESS OBJECT THROUGH PHYSICAL MAPS

Fig. 7A

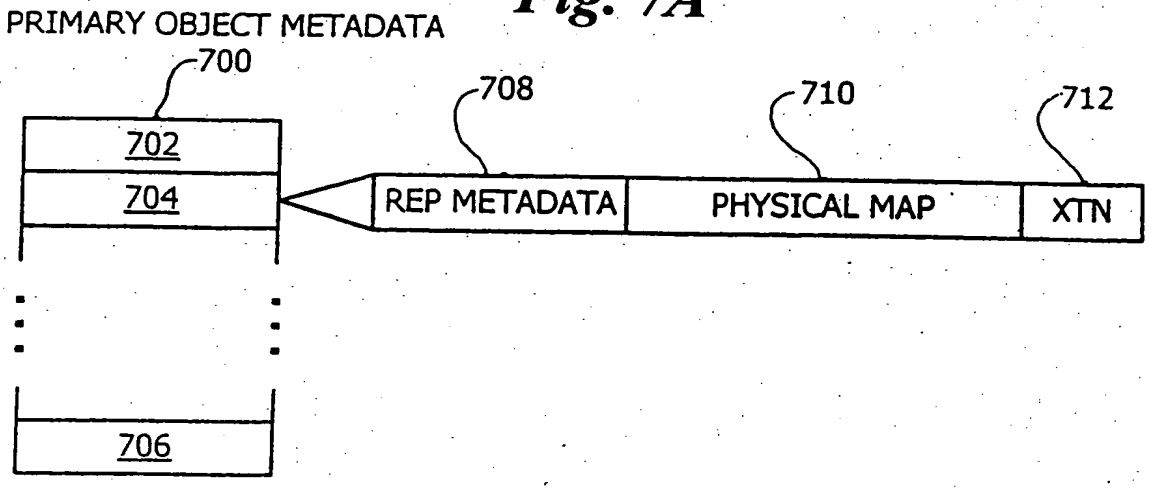


Fig. 7B

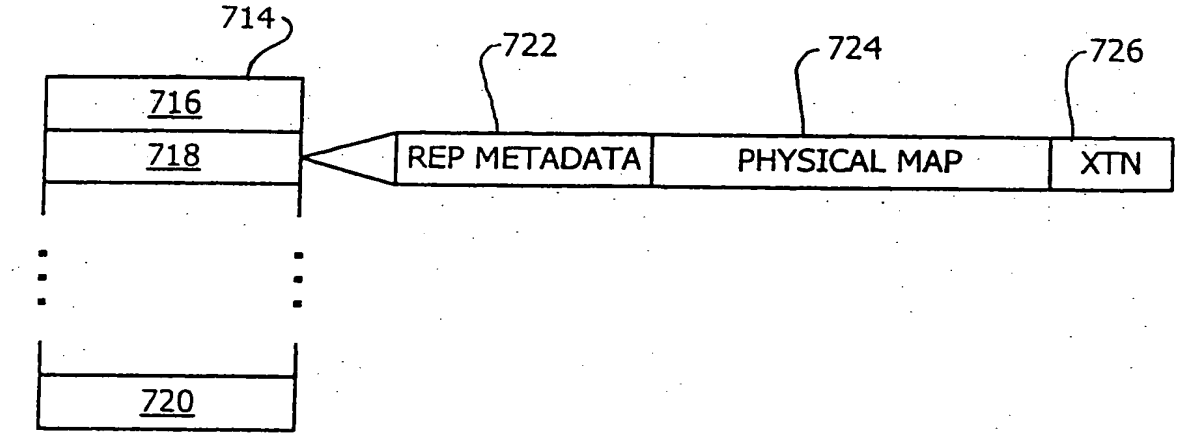


Fig. 7C

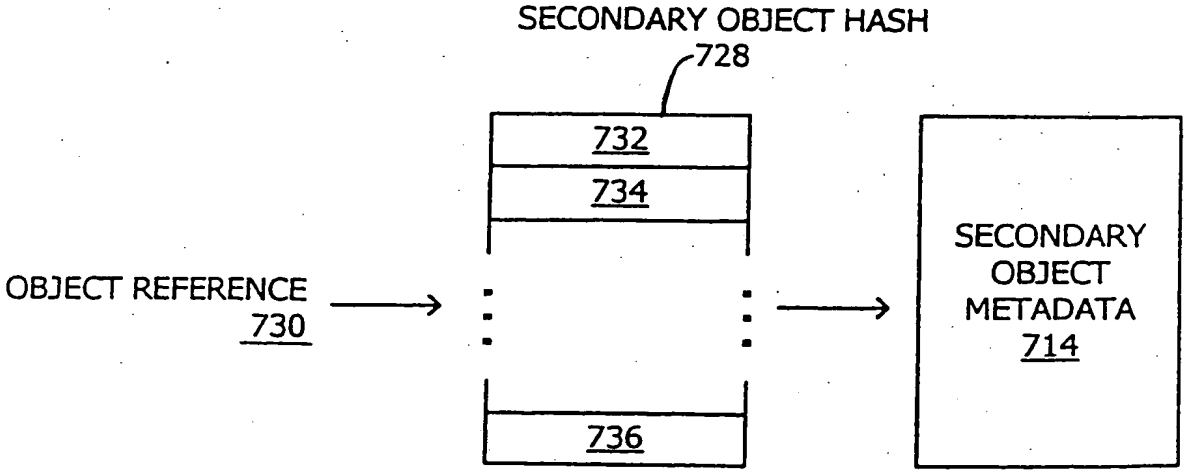


Fig. 8

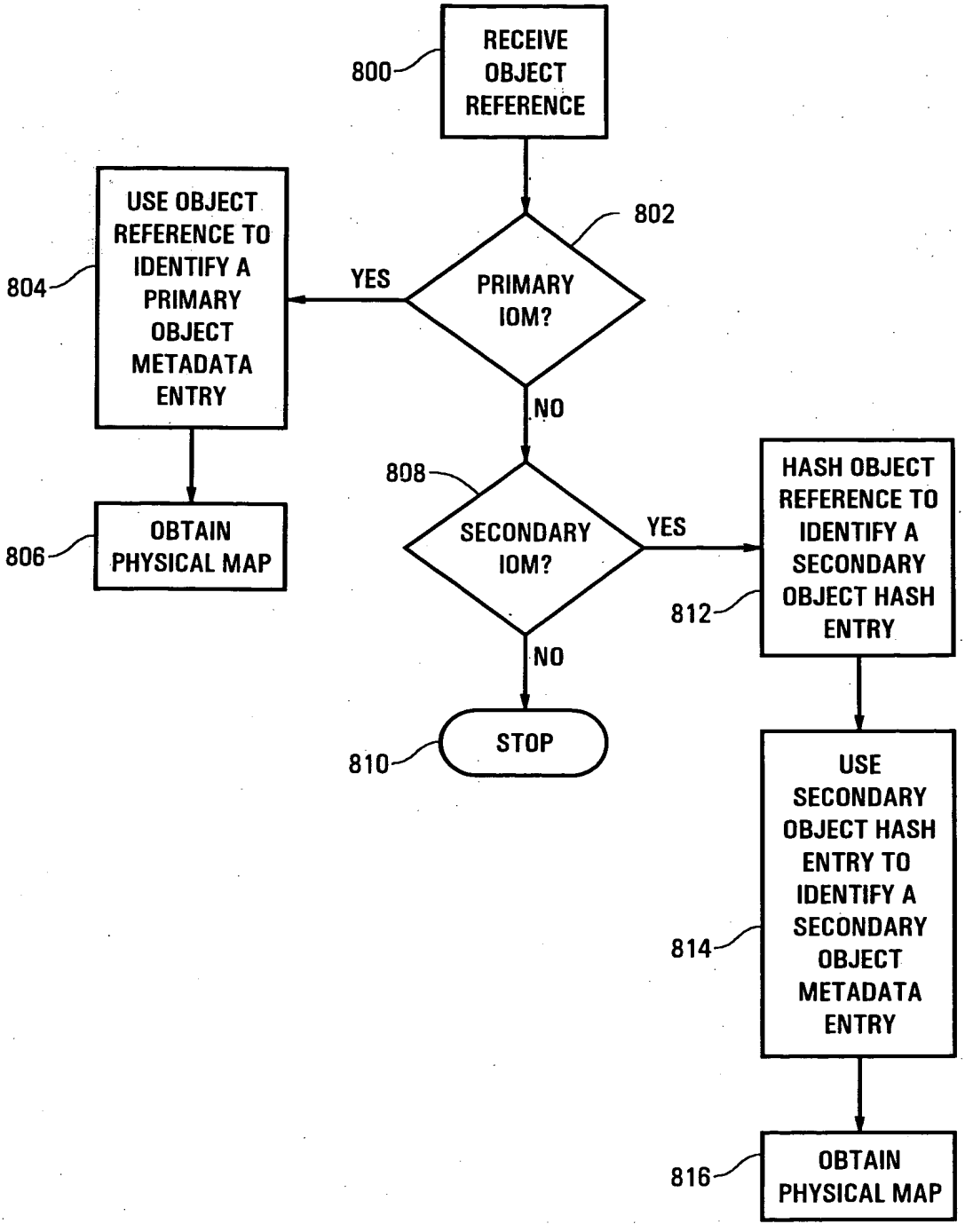

800
RECEIVE OBJECT REFERENCE
802
PRIMARY IOM?
YES
804
USE OBJECT REFERENCE TO IDENTIFY A PRIMARY OBJECT METADATA ENTRY
806
OBTAIN PHYSICAL MAP
NO
808
SECONDARY IOM?
YES
812
HASH OBJECT REFERENCE TO IDENTIFY A SECONDARY OBJECT HASH ENTRY
814
USE SECONDARY OBJECT HASH ENTRY TO IDENTIFY A SECONDARY OBJECT METADATA ENTRY
816
OBTAIN PHYSICAL MAP
NO
810
STOP

Fig. 9

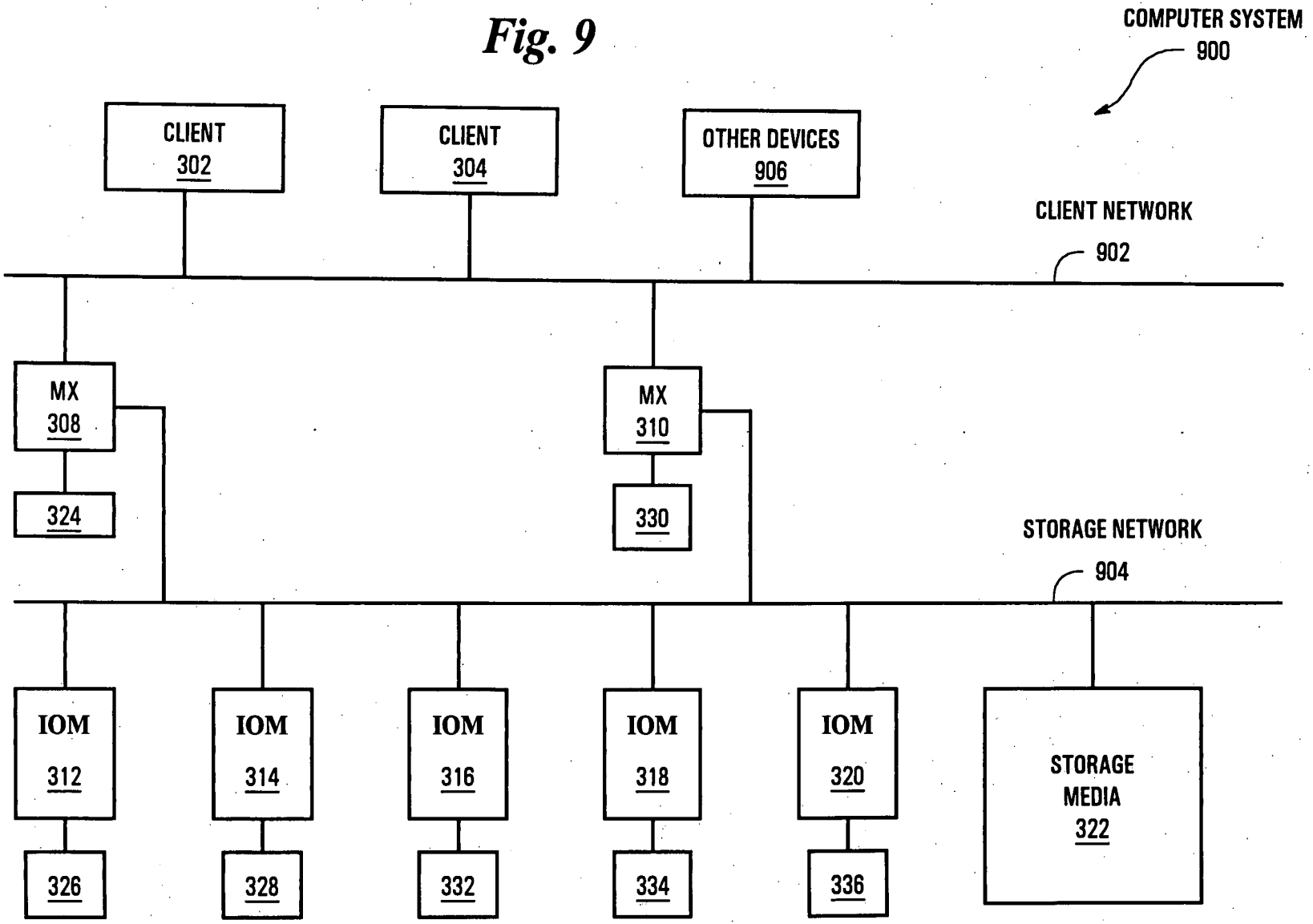

COMPUTER SYSTEM
900
CLIENT
302
CLIENT
304
OTHER DEVICES
906
CLIENT NETWORK
902
MX
308
MX
310
324
330
STORAGE NETWORK
904
IOM
312
IOM
314
IOM
316
IOM
318
IOM
320
STORAGE
MEDIA
322
326
328
332
334
336

Fig. 10

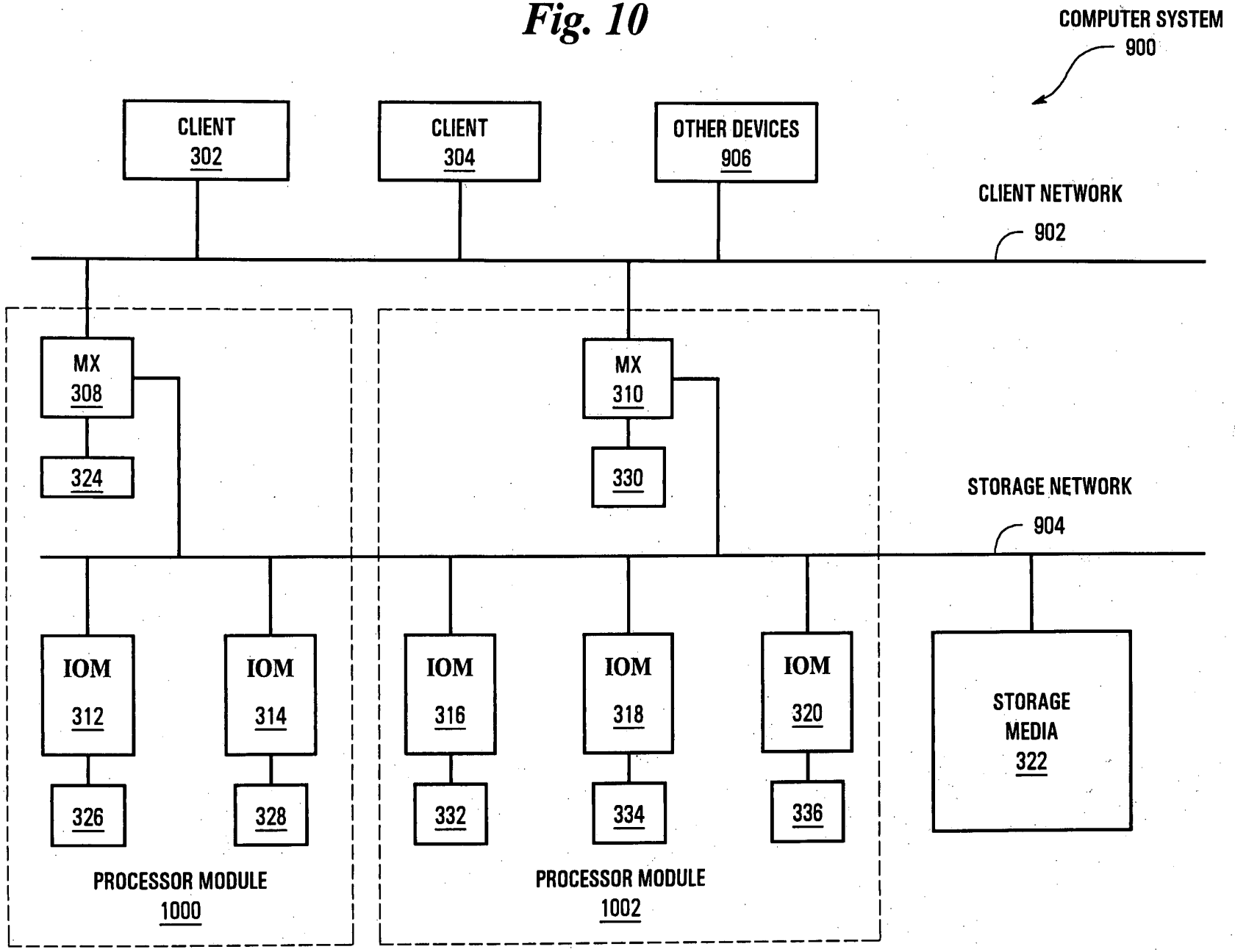

COMPUTER SYSTEM
900
CLIENT
302
CLIENT
304
OTHER DEVICES
906
CLIENT NETWORK
902
MX
308
324
MX
310
330
STORAGE NETWORK
904
IOM
312
IOM
314
IOM
316
IOM
318
IOM
320
STORAGE
MEDIA
322
326
328
332
334
336
PROCESSOR MODULE
1000
PROCESSOR MODULE
1002

Fig. 11

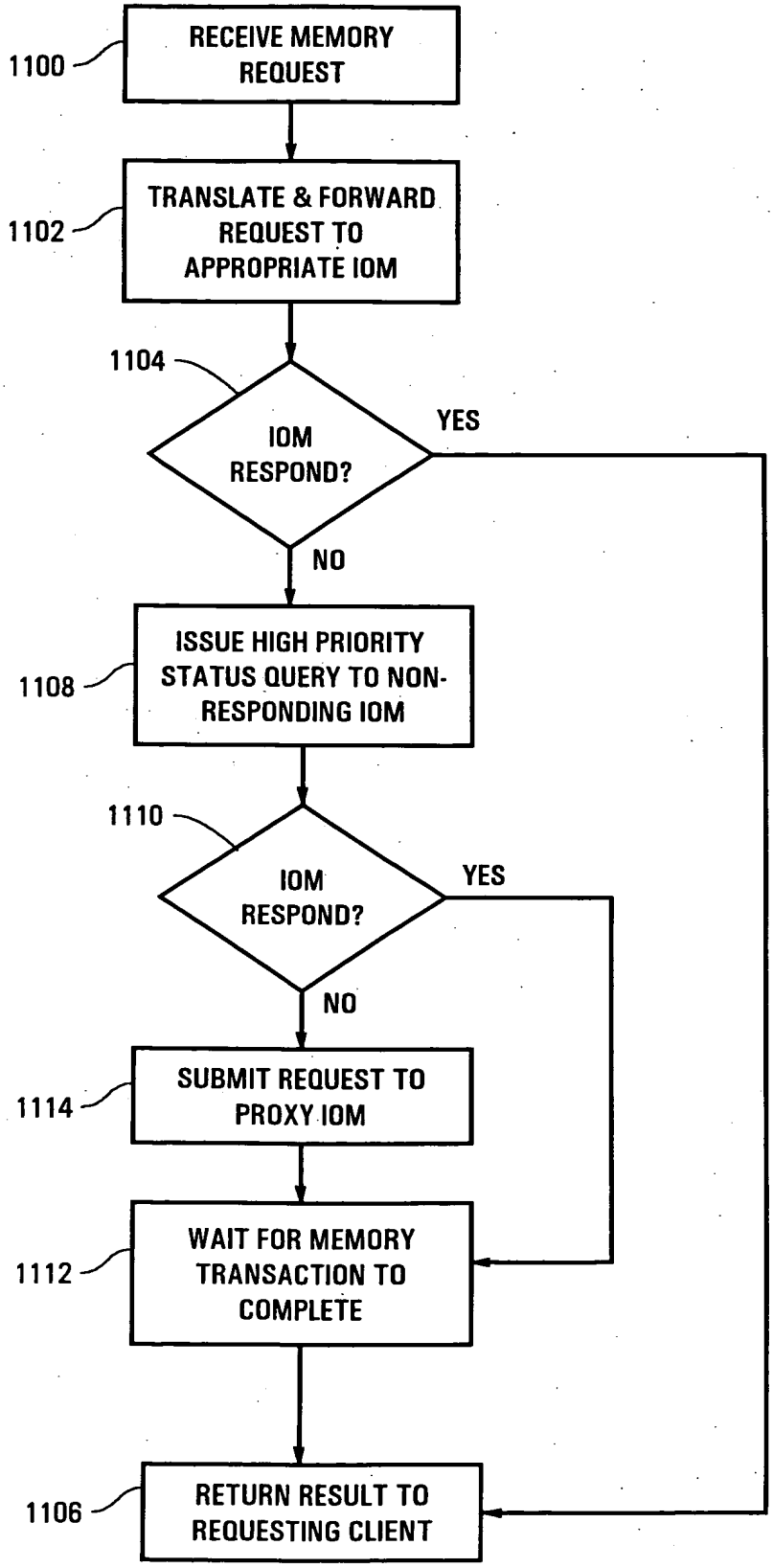

1100
RECEIVE MEMORY REQUEST
1102
TRANSLATE & FORWARD REQUEST TO APPROPRIATE IOM
1104
IOM RESPOND?
YES
NO
1108
ISSUE HIGH PRIORITY STATUS QUERY TO NON-RESPONDING IOM
1110
IOM RESPOND?
YES
NO
1114
SUBMIT REQUEST TO PROXY IOM
1112
WAIT FOR MEMORY TRANSACTION TO COMPLETE
1106
RETURN RESULT TO REQUESTING CLIENT

Fig. 12

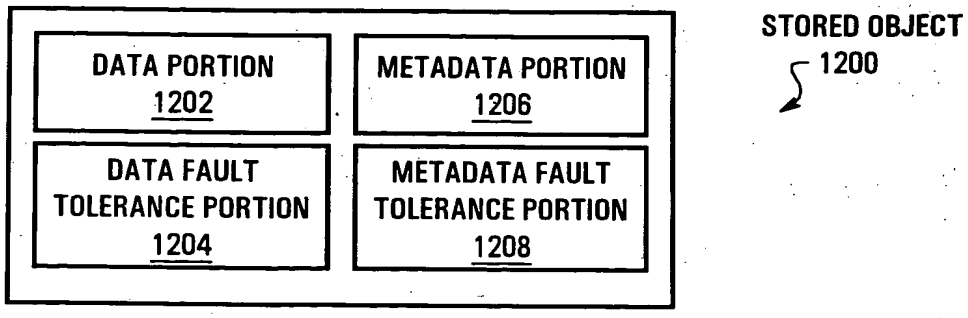

DATA PORTION
1202
METADATA PORTION
1206
DATA FAULT TOLERANCE PORTION
1204
METADATA FAULT TOLERANCE PORTION
1208
STORED OBJECT
1200

Fig. 13A

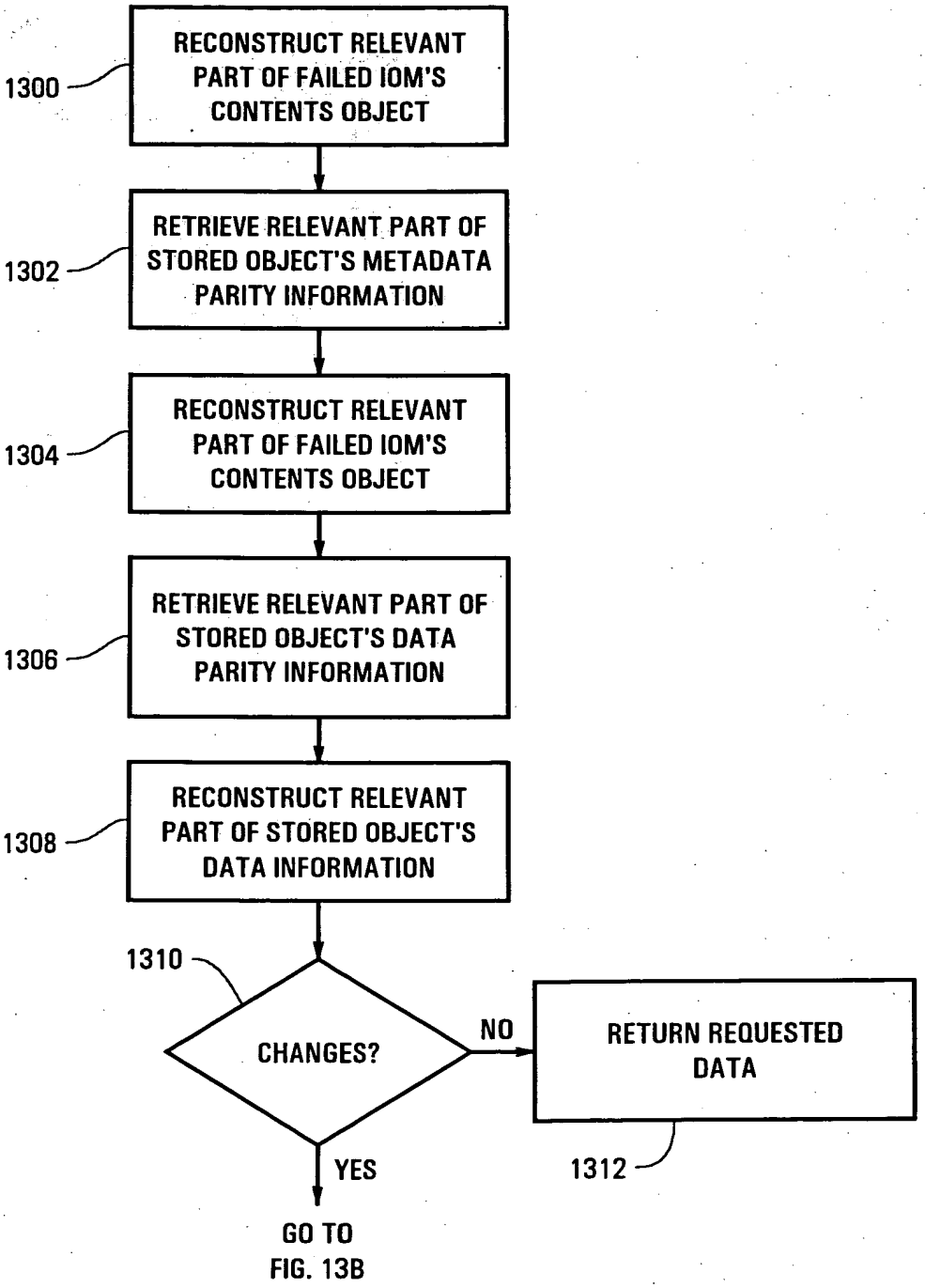
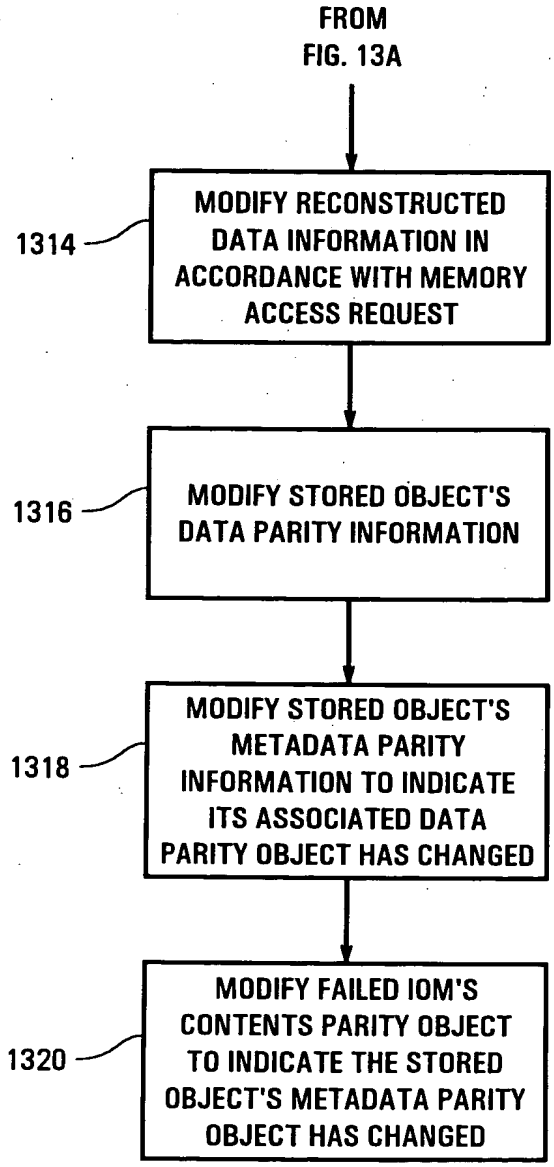

1300
RECONSTRUCT RELEVANT PART OF FAILED IOM'S CONTENTS OBJECT
1302
RETRIEVE RELEVANT PART OF STORED OBJECT'S METADATA PARITY INFORMATION
1304
RECONSTRUCT RELEVANT PART OF FAILED IOM'S CONTENTS OBJECT
1306
RETRIEVE RELEVANT PART OF STORED OBJECT'S DATA PARITY INFORMATION
1308
RECONSTRUCT RELEVANT PART OF STORED OBJECT'S DATA INFORMATION
1310
CHANGES?
NO
RETURN REQUESTED DATA
1312
YES
GO TO FIG. 13B

Fig. 13B


FROM
FIG. 13A
1314
MODIFY RECONSTRUCTED DATA INFORMATION IN ACCORDANCE WITH MEMORY ACCESS REQUEST
1316
MODIFY STORED OBJECT'S DATA PARITY INFORMATION
1318
MODIFY STORED OBJECT'S METADATA PARITY INFORMATION TO INDICATE ITS ASSOCIATED DATA PARITY OBJECT HAS CHANGED
1320
MODIFY FAILED IOM'S CONTENTS PARITY OBJECT TO INDICATE THE STORED OBJECT'S METADATA PARITY OBJECT HAS CHANGED

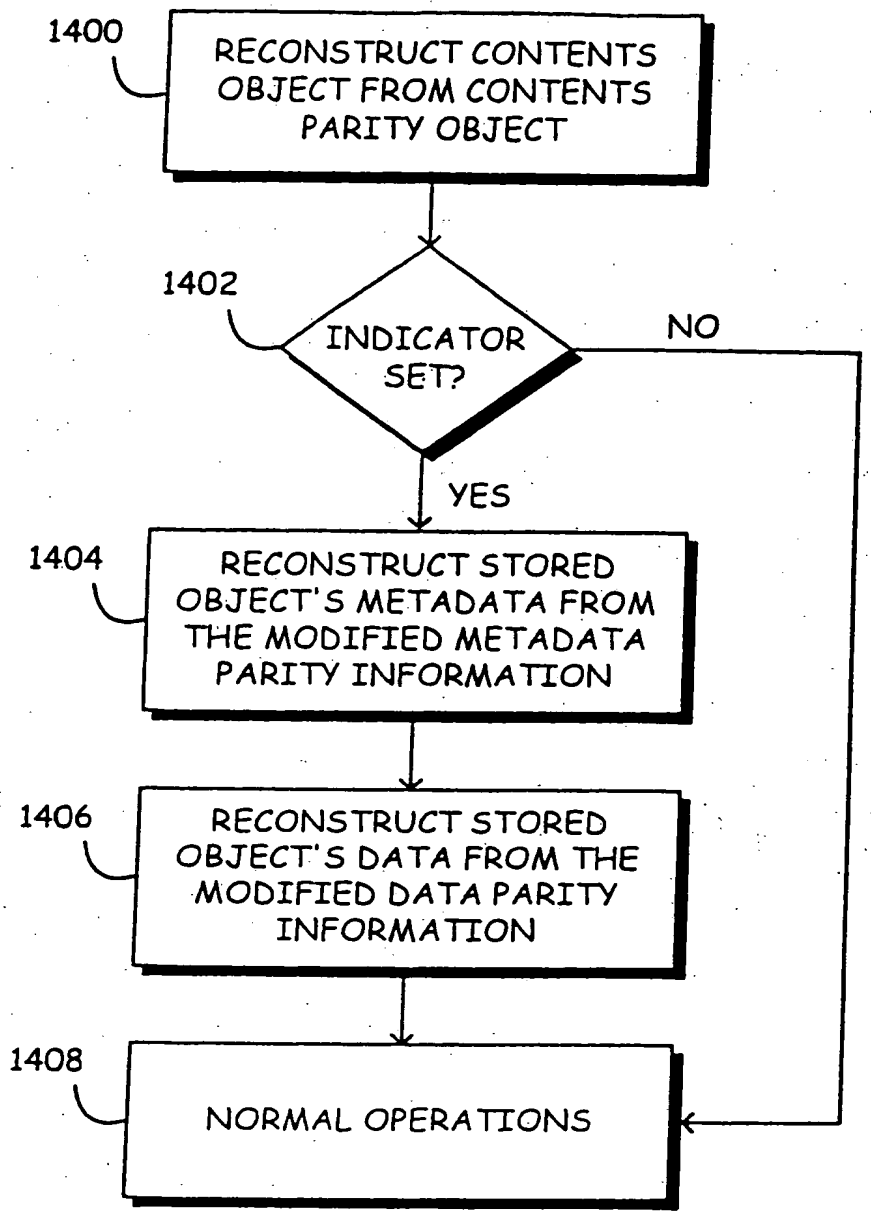

1400
RECONSTRUCT CONTENTS OBJECT FROM CONTENTS PARITY OBJECT
1402
INDICATOR SET?
NO
YES
1404
RECONSTRUCT STORED OBJECT'S METADATA FROM THE MODIFIED METADATA PARITY INFORMATION
1406
RECONSTRUCT STORED OBJECT'S DATA FROM THE MODIFIED DATA PARITY INFORMATION
1408
NORMAL OPERATIONS

FIG. 14

Fig. 15

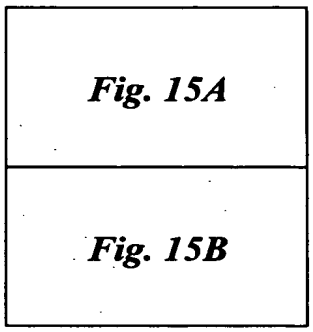


Fig. 15A

Time-line for a Data Write and Its Parity Update		
At the Data IOM	**Msg**	**At the Parity IOM**
Read old data with flags = DP_DONTBLOCK || PF_BC_READAHEAD || PF_BC_CREATE.		
Call modifyBlock getting a dirty buffer to hold new data transferred from caller (with a new disk address if needed).		
Make a journal entry for the write with a new ID = mj and the GFID, lblk, and new and old disk addresses for data.		
Transfer data into buffer. Set write_start and write_bytes to indicate the range of dirty bytes in the buffer that must be parity updated.		
Mark buffer tough and write data to disk with flags = DP_DONTBLOCK.		
Reply to caller if Toughness 0.		
Send ios_updatepn request to parity node with write_start, write_bytes, journal (mj & contents).	→	Receive ios_updatepn request.
Merge new data with old-data if necessary.		Get buffer for parity delta or data.
Compute parity delta or data (in a new buffer if necessary).		Write a new journal entry, pj, containing parity GFID, lblk, new and old disk addresses, and

Fig. 15B

		data of mj (i.e., mj, GFID, lblk, new and old disk address) with flags = DP_DONTBLOCK.
Receive request for parity delta or data, or the parity journal data.	←	Request new parity delta or data.
Return the parity delta. Then release the old data/delta buffer.	→	If needed, get old-parity with flags = DP_DONTBLOCK | PF_BC_READAHEAD
Reply to caller if Toughness 1.		Receive the parity delta in the new parity buffer.
When new data buffer is on disk, mark new data disk address in the cnode or indirect. Mark mj in their and the bitmap's pf_jrnrange.newest.		Merge the parity delta or data with the old parity buffer if needed. Then write new parity to disk with flags = DP_DONTBLOCK.
When mj and the new data buffer are on disk, send data hardened message to the parity node.	→	When new parity buffer is on disk, mark the new disk address and pj in pf_jrnrange.newest of the parity cnode or indirect buffer and of the bitmap.
Receive the hardened message from the parity node. If error, send update_pn to parity proxy.	←	Send parity update hardened message to the data node.
Reply to caller if Toughness 2.		Receive the hardened msg from the data node.
Deallocate the old data disk address if any and journal the mj_complete entry.		Finally, deallocate the old parity disk address if any and journal the pj_complete entry.

Fig. 16A

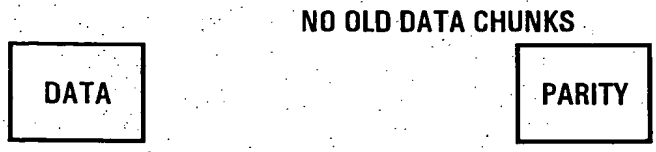


Fig. 16B

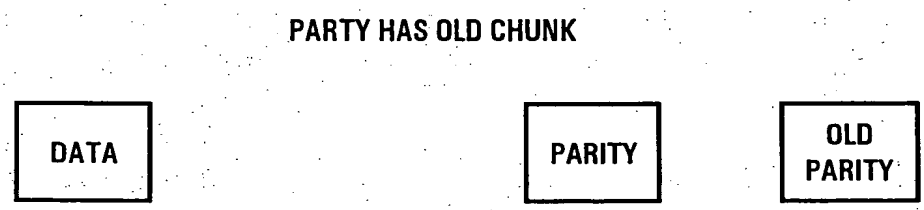


Fig. 16C

Fig. 16D